Package ‘bioCancer’

January 8, 2024

Title       Interactive Multi-Omics Cancers Data Visualization and Analysis
Version     1.30.0
Date        2023-10-19
Description This package is a Shiny App to visualize and analyse interactively Multi-Assays of Cancer Genomic Data.
Depends R (>= 3.6.0), radiant.data (>= 0.9.1), XML(>= 3.98)
Imports R.oo, R.methodsS3, httr, DT (>= 0.3), dplyr (>= 0.7.2), shiny (>= 1.0.5), AlgDesign (>= 1.1.7.3), import (>= 1.1.0), methods, AnnotationDbi, shinythemes, Biobase, geNetClassifier, org.Hs.eg.db, org.Bt.eg.db, DOSE, clusterProfiler, reactome.db, ReactomePA, DiagrammeR(<= 1.01), visNetwork, htmlwidgets, plyr, tibble, GO.db
Suggests BiocStyle, prettydoc, rmarkdown, knitr, testthat (>= 0.10.0)
VignetteBuilder knitr
URL         https://kmezhoud.github.io/bioCancer/
BugReports  https://github.com/kmezhoud/bioCancer/issues
License     AGPL-3 | file LICENSE
LazyData    true
biocViews   GUI, DataRepresentation, Network, MultipleComparison, Pathways, Reactome, Visualization, GeneExpression, GeneTarget
RoxygenNote 7.2.3
Encoding    UTF-8
git_url     https://git.bioconductor.org/packages/bioCancer
git_branch  RELEASE_3_18
git_last_commit  54bba2e
git_last_commit_date  2023-10-24
Repository  Bioconductor 3.18
Date/Publication 2024-01-08
Author      Karim Mezhoud [aut, cre]
Maintainer  Karim Mezhoud <kmezhoud@gmail.com>
R topics documented:

AnnotationFuncs-package .................................................. 3
.dbEscapeString ............................................................... 4
.getTableNames ............................................................... 5
.pickRef ................................................................. 5
.attriColorGene ......................................................... 6
.attriColorValue ......................................................... 7
.attriColorVector ......................................................... 8
.attriShape2Gene ......................................................... 8
.attriShape2Node ......................................................... 9
.bioCancer ................................................................. 10
.CGDS ................................................................. 10
.checkDimensions ......................................................... 11
.coffeewheel ............................................................... 11
coffeewheelOutput ...................................................... 12
displayTable ............................................................... 13
.Edges_Diseases_obj .................................................... 13
epiGenomics ............................................................... 14
.findPhantom ............................................................... 15
.getCancerStudies.CGDS .................................................. 15
.getCaseLists.CGDS ....................................................... 16
.getClinicalData.CGDS .................................................... 16
.getEvidenceCodes ....................................................... 17
.getFreqMutData ............................................................ 18
.getGenesClassification .................................................. 18
.getGeneticProfiles.CGDS ................................................ 19
.getListProfData ........................................................... 20
.getList_Cases ............................................................. 21
.getList_GenProfs .......................................................... 21
.getMegaProfData .......................................................... 22
.getMutationData.CGDS ................................................... 23
.getOrthologs ............................................................... 24
.getProfileData.CGDS ..................................................... 25
getSequensed_SampleSize ................................................ 26
grepRef ................................................................. 27
.mapLists ................................................................. 28
.metabologram ............................................................. 29
.metabologramOutput ...................................................... 30
.mutation_obj ............................................................. 30
.Node_df_FreqIn ............................................................ 31
.Node_Diseases_obj ........................................................ 32
.Node_obj_CNA_ProfData .................................................. 32
.Node_obj_FreqIn .......................................................... 33
.Node_obj_Met_ProfData ................................................... 34
.Node_obj_mRNA_Classifier ............................................... 34
.pickGO ................................................................. 35
.pickRefSeq ............................................................... 37
Annotation Funcs-package

Description

Package: AnnotationFuncs
Type: Package
Version: 1.3.0
Date: 2011-06-10
License: GPL-2
LazyLoad: yes

Details

Functions for handling translations between different identifiers using the Biocore Data Team data-packages (e.g. org.Bt.EG.db). Primary functions are translate for translating and getOrthologs for efficient lookup of homologues using the Inparanoid databases. Other functions include functions for selecting Refseqs or Gene Ontologies (GO).
Author(s)
Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>

References

See Also
translate, getOrthologs

Examples
library(org.Bt.eg.db)
gene.symbols <- c('DRBP1','SERPINA1','FAKE','BLABLA')
# Find entrez identifiers of these genes.
eg <- translate(gene.symbols, org.Bt.egSYMBOL2EG)
# Note that not all symbols were translated.

# Go directly to Refseq identifiers.
refseq <- translate(gene.symbols, from=org.Bt.egSYMBOL2EG, to=org.Bt.egREFSEQ)
# Pick the proteins:
pickRefSeq(refseq, priorities=c('NP','XP'), reduce='all')

.dbEscapeString  Private Escape string

Description
Does not escape strings, but raises an error if any character expect normal letters and underscores are found in the string.

Usage
.dbEscapeString(str, raise.error = TRUE)

Arguments
str    String to test
raise.error Logical, whether to raise an error or not.

Value
Invisible logical
.getTableName

*Gets the table name from the INPARANOID style genus names.*

**Description**

Gets the table name from the INPARANOID style genus names.

**Usage**

```r
.getTableName(genus)
```

**Arguments**

- **genus**
  - 5 character INPARANOID genus name, such as "BOSTA", "HOMSA" or "MUSMU".

**Value**

Table name for genus.

**Author(s)**

Stefan McKinnon Edwards <stefanm.edwards@agrsci.dk>

**References**

[https://www.bioconductor.org/packages/release/bioc/html/AnnotationDbi.html](https://www.bioconductor.org/packages/release/bioc/html/AnnotationDbi.html)

---

.pickRef

*Secret function that does the magic for pickRefSeq.*

**Description**

Do not use it, use `pickRefSeq`!

**Usage**

```r
.pickRef(l, priorities, reduce = c("all", "first", "last"))
```

**Arguments**

- **l**
  - List.
- **priorities**
  - How to prioritize.
- **reduce**
  - How to reduce.
attriColorGene

Value

List.

Note

Hey, you found a secret function! Keep it that way!

Author(s)

Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>

See Also

pickRefSeq

attriColorGene | Attribute Color to Gene

Description

Attribute Color to Gene

Usage

attriColorGene(df)

Arguments

df | data frame with mRNA or CNA or mutation frequency or methylation (numeric).

Value

A list colors for every gene

Examples

```r
## Not run:
library(CGDS)
cgds <- CGDS("http://www.cbioportal.org/")
genList <- whichGeneList("73")
ProfData <- getProfileData.CGDS(cgds,
                                genList, "gbm_tcgag_pub_mrna", "gbm_tcgag_pub_all")
rownames(ProfData) <- NULL
clr <- attriColorGene(ProfData)
## End(Not run)
```
Description

Attribute Color to Value

Usage

attriColorValue(Value, df, colors=c(a,b,c),feet)

Arguments

Value  integer
df      data frame with numeric values
colors  a vector of 5 colors
feet    the interval between two successive colors in the palette (0.1)

Value

Hex Color Code

Examples

## Not run:
cgds <- CGDS("http://www.cbiportal.org/")
genelist <- whichGeneList("73")
ProfData <- getProfileData.CGDS(cgds,
  genelist, "gbm_tcgagp_pub_mrna", "gbm_tcgagp_pub_all")
rownames(ProfData) <- NULL
clrRef <- attriColorValue(1.2,
  ProfData,
  colors = c("blue3", "white","red"),
  feet=10)

## End(Not run)
attriColorVector  

Attribute color to a vector of numeric values

Description  
Attribute color to a vector of numeric values

Usage  

attriColorVector(Value, vector, colors=c(a,b,c),feet)

Arguments  

Value  numeric  
vector  A vector of numeric data  
colors  3 colors  
feet  An interval between two numeric value needed to change the color

Value  
A vetor of colors

Examples  

## Not run:  
cgds <- CGDS("http://www.cbiportal.org/")  
genList <- whichGeneList("73")  
ProfData <- getProfileData.CGDS(cgds,  
genList, "gbm_tcgag_pub_mrna", "gbm_tcgag_pub_all")  
rownames(ProfData) <- NULL  
clrVec <- attriColorVector(1.2,  
ProfData[1,],  
colors = c("blue", "white", "red"),  
feet=1)

## End(Not run)

attriShape2Gene  

Attribute shape to nodes

Description  
Attribute shape to nodes

Usage  

attriShape2Gene(gene, genelist)
attriShape2Node

Arguments

gene Gene symbol
genelist Gene list

Value

A character "BRCA1[shape = 'circle',"

Examples

how <- "runManually"
## Not run:
GeneList <- whichGeneList("73")
attriShape2Gene("P53", GeneList)
attriShape2Gene("GML", GeneList)

## End(Not run)

attriShape2Node

Attributes shape to Nodes

Description

Attributes shape to Nodes

Usage

attriShape2Node(gene, genelist)

Arguments

gene symbol "TP53"
genelist a vector of gene symbol

Value

A data frame with egdes attributes

Examples

GeneList <- c("DKK3", "NBN", "MYO6", "TP53", "PML", "IFI16", "BRCA1")
NodeShape <- attriShape2Gene("DKK3", GeneList)
bioCancer

Launch bioCancer with default browser

Description

The Main function to run bioCancer App

Usage

bioCancer()

Value

web page of bioCancer Shiny App

Examples

ShinyApp <- 1
## Not run:
bioCancer()
## End(Not run)

CGDS

CGDS connect object to cBioPortal

Description

Creates a CGDS connection object from a CGDS endpoint URL. This object must be passed on to the methods which query the server.

Usage

CGDS(url,verbose=FALSE,ploterrormsg='',token=NULL)

Arguments

url A CGDS URL (required).
verbose A boolean variable specifying verbose output (default FALSE)
ploterrormsg An optional message to display in plots if an error occurs (default ”)
token An optional ’Authorization: Bearer’ token to connect to cBioPortal instances that require authentication (default NULL)
checkDimensions

Description

Check which Cases and genetic profiles are available for every selected study.

Usage

checkDimensions(panel, StudyID)

Arguments

panel panel can take to strings 'Circomics' or 'Networking'
StudyID Study reference using cBioPortal index

Value

A data frame with two column (Cases, Genetic profiles). Every row has a dimension (CNA, mRNA...). The data frame is filled with yes/no response.

Examples

## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
df <- checkDimensions(panel='Networking', StudyID= "gbm_tcga_pub")
## End(Not run)

coffeewheel

This is an htmlwidgets-based visualization tool for hierarchical data. It is zoomable, meaning that you can interact with the hierarchy and zoom in/out accordingly.

Description

This is an htmlwidgets-based visualization tool for hierarchical data. It is zoomable, meaning that you can interact with the hierarchy and zoom in/out accordingly.

Usage

coffeewheel(treeData, width=600, height=600, main="", partitionAttribute="value")
coffeewheelOutput

Arguments

- treeData: A hierarchical tree data as in example
- width: 600
- height: 600
- main: Title
- partitionAttribute: "value"

Value

A circular layout with genetic profile.

Examples

How <- "runManually"
## Not run:
coffeewheel(treeData = sampleWheelData)
## End(Not run)

coffeewheelOutput  Widget output function for use in Shiny

Description

Widget output function for use in Shiny

Usage

coffeewheelOutput(outputId, width=700, height=700)

Arguments

- outputId: id
- width: 700
- height: 700

Value


Examples

How <- "runManually"
## Not run:
coffeewheel(treeData = sampleWheelData)
## End(Not run)
displayTable

Display dataframe in table using DT package

Description
Display dataframe in table using DT package

Usage
displayTable(df)

Arguments
  df  a dataframe

Value
A table

Examples
## Not run:
session <- NULL
cgds <- CGDS("http://www.cbioportal.org/")
Studies<- getCancerStudies.CGDS(cgds)
displayTable(Studies)
## End(Not run)

Edges_Diseases_obj get Edges dataframe for Gene/Disease association from geNetClassifier

Description
get Edges dataframe for Gene/Disease association from geNetClassifier

Usage
Edges_Diseases_obj(genesclassdetails)

Arguments
  genesclassdetails
  a dataframe from geNetClassifier
Value

A data frame with edges attributes

Examples

```r
GenesClassDetails <- structure(list(Genes = c("FANCF", "MLH1", "MSH2", "ATR", "PARP1", "CHEK2", "RAD51"), ranking = c(1L, 1L, 1L, 2L, 3L, 1L, 2L), class = c("brca_tcga", "gbm_tcga", "lihc_tcga", "lihc_tcga", "lihc_tcga", "lusc_tcga", "lusc_tcga"), postProb = c(1, 0.99, 1, 0.99, 0.99, 1, 0.98), exprsMeanDiff = c(180, 256, -373, -268, -1482, 258, 143), exprsUpDw = c("UP", "UP", "DOWN", "DOWN", "UP", "UP")), .Names = c("Genes", "ranking", "class", "postProb", "exprsMeanDiff", "exprsUpDw"), class = "data.frame", row.names = c(NA,-7L))

Ed_Diseases_obj <- Edges_Diseases_obj(genesclassdetails=GenesClassDetails)
```

Description

Default dataset of bioCancer

Usage

epiGenomics

Format

An object of class data.frame with 48 rows and 7 columns.

Author(s)

Karim Mezhoud <kmezhoud@gmail.com>
findPhantom

Check if PhantomJS is installed. Similar to webshot

Description

Check if PhantomJS is installed. Similar to webshot

Usage

findPhantom()

Value

Logic object

Examples

## Not run:
findPhantom()
## End(Not run)

getCancerStudies.CGDS

S3 method to get Cancer Studies

Description

S3 method to get Cancer Studies

Usage

## S3 method for class 'CGDS'
getCancerStudies(x, ...)

Arguments

x  connection object
...
  not used

Examples

# Create CGDS object
mycgds <- CGDS("http://www.cbioportal.org/")
# Get available case lists (collection of samples) for a given cancer study
mycancerstudy <- getCancerStudies.CGDS(mycgds)[2,1]
getCaseLists.CGDS  
**S3 method to get Cases Lists**

**Description**
S3 method to get Cases Lists

**Usage**
```r
## S3 method for class 'CGDS'
getCaseLists(x, cancerStudy, ...)
```

**Arguments**
- `x`  
  - connection object
- `cancerStudy`  
  - cancer study ID
- `...`  
  - Not used

**Examples**
```r
# Create CGDS object
mycgds <- CGDS("http://www.cbioportal.org/")
# Get list of cancer studies at server
mycancerstudy <- getCancerStudies.CGDS(mycgds)[2,1]
# Get available case lists (collection of samples) for a given cancer study
mycaselist <- getCaseLists.CGDS(mycgds,mycancerstudy)[1,1]
```

---

getClinicalData.CGDS  
**S3 method to get Clinical Data**

**Description**
S3 method to get Clinical Data

**Usage**
```r
## S3 method for class 'CGDS'
getClinicalData(x, caseList = "", cases = c(), caseIdsKey = "", ...)
```

**Arguments**
- `x`  
  - connection object
- `caseList`  
  - A list of cases ID
- `cases`  
  - A vector of case IDs
- `caseIdsKey`  
  - only used by web portal
- `...`  
  - not used
getEvidenceCodes

Examples

# Create CGDS object
mycgds <- CGDS("http://www.cbioportal.org/")
# Get available case lists (collection of samples) for a given cancer study
mycancerstudy <- getCancerStudies.CGDS(mycgds)[2,1]
mycaselist <- getCaseLists.CGDS(mycgds,mycancerstudy)[1,1]

getEvidenceCodes

Returns GO evidence codes.

Description

Returns GO evidence codes.

Usage

getEvidenceCodes()

Value

Matrix of two columns, first column with codes, second column with description of codes.

Author(s)

Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>

References

?org.Bt.egGO

See Also

pickGO

Examples

getEvidenceCodes()
getFreqMutData

**Description**

get mutation frequency

**Usage**

getFreqMutData(list, geneListLabel)

**Arguments**

- `list`: a list of data frame with mutation data. Each data frame is for one study
- `geneListLabel`: file name of geneList examples: "73"

**Value**

a data frame with mutation frequency. gene is in rows and study is in column

**Examples**

```r
## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
genelist <- whichGeneList("73")
r_data <- new.env()
MutData <- getMutationData.CGDS(cgds,"gbm_tcga_pub_all",
  "gbm_tcga_pub_mutations", genelist )
FreqMut <- getFreqMutData(list(ls1=MutData, ls2=MutData), "73")
## End(Not run)
```

getGenesClassification

**Description**

get genes classification

**Usage**

getGenesClassification(checked_Studies, GeneList, samplesize, threshold, listGenProfs, listCases)
getGeneticProfiles.CGDS

Arguments

- "checked_Studies" checked studies
- "GeneList" gene list
- "samplesize" sample size
- "threshold" p-value threshold
- "listGenProfs" list of genetic profiles
- "listCases" list of cases

Value

A table with genes classed by study

Examples

```r
## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
listStudies <- getCancerStudies.CGDS(cgds)
checked_Studies <- listStudies[3:5]
listCases <- getList_Cases(listStudies[1:3])
listGenProfs <- getList_GenProfs(listStudies[1:3])
GeneList <- c("P53", "IFI16", "BRCA1")
samplesize <- 50
threshold <- 0.95
table <- getGenesClassification(checked_Studies, GeneList,
samplesize, threshold, listGenProfs, listCases)
## End(Not run)
```

---

generateProfiles.CGDS

_S3 method to get Genetic Profiles_

Description

S3 method to get Genetic Profiles

Usage

```r
## S3 method for class 'CGDS'
getGeneticProfiles(x, cancerStudy, ...)
```

Arguments

- "x" connection object
- "cancerStudy" cancer study ID
- "..." not used
Examples

# Create CGDS object
mycgds <- CGDS("http://www.cbioportal.org/")
# Get list of cancer studies at server
mycancerstudy <- get CancerStudies.CGDS(mycgds)[2,1]
# Get available case lists (collection of samples) for a given cancer study
mycaselist <- get CaseLists.CGDS(mycgds, mycancerstudy)[1,1]
# Get available genetic profiles
mygeneticprofile <- get GeneticProfiles.CGDS(mycgds, mycancerstudy)[1,1]
# Get data slices for a specified list of genes, genetic profile and case list
myProfileData <- getProfileData.CGDS(mycgds, c('BRCA1', 'BRCA2'), mygeneticprofile, mycaselist)

---

**getListProfData**

get list of data frame with profiles data (CNA, mRNA, Methylation, Mutation...)

Description

get list of data frame with profiles data (CNA, mRNA, Methylation, Mutation...)

Usage

getListProfData(panel, geneListLabel)

Arguments

panel Panel name (string) in which Studies are selected. There are two panels ("Circomics" or "Networking")

geneListLabel The label of GeneList. There are three cases: "Genes" user gene list, "Reactome_GeneList" GeneList plus genes from reactomeFI "file name" from Examples

Value

A LIST of profiles data (CNA, mRNA, Methylation, Mutation, miRNA, RPPA). Each dimension content a list of studies.

Examples

```
## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
geneList <- whichGeneList("73")
r_data <- new.env()
MutData <- getMutationData.CGDS(cgds,"gbm_tcga_pub_all","gbm_tcga_pub_mutations", geneList )
FreqMut <- getFreqMutData(list(ls1=MutData, ls2=MutData), "73")
input <- NULL
```
**getList_Cases**

`getList_Cases(checked_Studies)`

**Description**

get list of cases of each selected study in Classifier panel

**Usage**

```r
getList_Cases(checked_Studies)
```

**Arguments**

checked_Studies
cHECKED studies

**Value**

d List of cases

**Examples**

```r
## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
listStudies <- getCancerStudies.CGDS(cgds)
listCases <- getList_Cases(listStudies[1:3])
## End(Not run)
```

**getList_GenProfs**

`getList_GenProfs(checked_Studies)`

**Description**

get list of genetic profiles of each selected study in Classifier panel

**Usage**

```r
getList_GenProfs(checked_Studies)
```
getMegaProfData

Arguments

- **checked_Studies**
  - checked studies

Value

- listes of genetics profiles

Examples

```r
## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
listStudies <- getCAncerStudies.CGDS(cgds)
listGenProfs <- getList_GenProfs(listStudies[1:3])

## End(Not run)
```

getMegaProfData  search and get genetic profiles (CNA, mRNA, Methylation, Mutation...) of gene list upper than 500

Description

search and get genetic profiles (CNA, mRNA, Methylation, Mutation...) of gene list upper than 500

Usage

getMegaProfData(MegaGeneList, GenProf, Case, Class)

Arguments

- **MegaGeneList**
  - A list of genes upper than 500
- **GenProf**
  - genetic profile reference
- **Case**
  - Case reference
- **Class**
  - indicates the panel ProfData or Mutdata

Details

See https://github.com/kmezhoud/bioCancer/wiki

Value

- A data frame with Genetic profile
**Examples**

```r
GeneList <- c("ALK", "JAK3", "SHC3", "TP53", "MYC", "PARP")
## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
listCase_gbm_tcga_pub <- getCaseLists.CGDS(cgds,"gbm_tcga_pub")[,1]
listGenProf_gbm_tcga_pub <- getGeneticProfiles.CGDS(cgds,"gbm_tcga_pub")[,1]

ProfData_Mut <- grepRef("gbm_tcga_pub_all", listCase_gbm_tcga_pub, "gbm_tcga_pub_mutations", listGenProf_gbm_tcga_pub, GeneList, Mut=1)

## End(Not run)
```

---

**getMutationData.CGDS  S3 method to ge Mutation Data**

**Description**

S3 method to ge Mutation Data

**Usage**

```r
## S3 method for class 'CGDS'
getMutationData(x, caseList, geneticProfile, genes, ...)
```

**Arguments**

- `x` : connection object
- `caseList` : A case list ID
- `geneticProfile` : A genetic profile ID with mutation data
- `genes` : A vector of genes list
- `...` : not used

**Examples**

```r
#Create CGDS object
mycgds <- CGDS("http://www.cbioportal.org/")
# Get Extended Mutation Data for EGFR and PTEN in TCGA GBM
myMutationData <- getMutationData.CGDS(mycgds,"gbm_tcga_all","gbm_tcga_mutations", c('EGFR','PTEN'))
```
getOrthologs

Performs quicker lookup for orthologs in homologe data packages

Description

Using the INPARANOID data packages such as hom.Hs.inp.db is very, very slow and can take up to 11 min (on this particular developers workstation). This function introduces a new method that can do it in just 20 seconds (on the developers workstation). In addition, it includes options for translating between different identifiers both before and after the mapping.

Usage

getOrthologs(
  values,
  mapping,
  genus,
  threshold = 1,
  pre.from = NULL,
  pre.to = NULL,
  post.from = NULL,
  post.to = NULL,
  ...
)

Arguments

values  Vector, coerced to character vector, of values needed mapping by homology.
mapping Homology mapping object, such as hom.Hs.inpBOSTA or revmap(hom.Hs.inpBOSTA).
genus Character vector. 5 character INPARANOID style genus name of the mapping object, e.g. 'BOSTA' for both hom.Hs.inpBOSTA and revmap(hom.Hs.inpBOSTA).
threshold Numeric value between 0 and 1. Only clustered homologues with a pairwise score above the threshold is included. The native implementation has this set to 1.
pre.from Mapping object if values needs translation before mapping. E.g. values are entrez and hom.Hs.inpBOSTA requires ENSEMBLPROT, hom.Hs.inpAPIME requires Refseq (?). Arguments from and to are just like in translate.
pre.to Second part of translation before mapping.
post.from Translate the result from homology mapping to a desired id; just like in translate.
post.to Second part of translation after mapping.
... Additional arguments sent to translate.

Value

List. Names of list corresponds to values, except those that could not be mapped nor translated. Entries are character vectors.
Author(s)

Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>

References

?hom.Hs.inp.db - https://inparanoidb.sbc.su.se/


See Also

translate, .getTableName, mapLists

Examples

tmp <- 1

getProfileData.CGDS S3 method to get Profile Data

Description

S3 method to get Profile Data

Usage

## S3 method for class 'CGDS'
getProfileData(
  x,
  genes,
  geneticProfiles,
  caseList = "",
  cases = c(),
  caseIdsKey = "",
  ...
)
Arguments

- `x` connection object
- `genes` A genes list
- `geneticProfiles` A genetic Profile ID
- `caseList` A cases list ID
- `cases` A vector of cases ID
- `caseIdsKey` Only used by web portal
- `...` not used

Examples

```r
# Create CGDS object
mycgds <- CGDS("http://www.cbiportal.org/")
# Get list of cancer studies at server
mycancerstudy <- getCancerStudies.CGDS(mycgds)[2,1]
# Get available case lists (collection of samples) for a given cancer study
mycaselist <- getCaseLists.CGDS(mycgds,mycancerstudy)[1,1]
# Get available genetic profiles
mygeneticprofile <- getGeneticProfiles.CGDS(mycgds,mycancerstudy)[1,1]
# Get data slices for a specified list of genes, genetic profile and case list
myProfileData <- getProfileData.CGDS(mycgds,c('BRCA1','BRCA2'),mygeneticprofile,mycaselist)
# Get data slice for a single gene
mysigneProfileData <- getProfileData.CGDS(mycgds,'HMGA2',mygeneticprofile,mycaselist)
```

getSequensed_SampleSize

`getSequensed_SampleSize` is a function to get the sample size of sequenced genes.

Description

get samples size of sequenced genes

Usage

getSequensed_SampleSize(StudyID)

Arguments

- `StudyID` Study reference using cBioPortal index

Value

dataframe with sample size for each selected study.
grepRef

## Not run:
```r
sampleSize <- getSequensed_SampleSize(input$StudiesIDCircos)
```
## End(Not run)

---

grepRef

### Description

search and get genetic profiles (CNA, mRNA, Methylation, Mutation...)

### Usage

```r
grepRef(regex1, listRef1, regex2, listRef2, GeneList, Mut)
```

### Arguments

- **regex1**: Case id (cancer_study_id_[mutations, cna, methylation, mrna]).
- **listRef1**: A list of cases for one study.
- **regex2**: Genetic Profile id (cancer_study_id_[mutations, cna, methylation, mrna]).
- **listRef2**: A list of Genetic Profiles for one study.
- **GeneList**: A list of genes
- **Mut**: Condition to set if the genetic profile is mutation or not (0,1)

### Details

See [https://github.com/kmezhoud/bioCancer/wiki](https://github.com/kmezhoud/bioCancer/wiki)

### Value

A data frame with Genetic profile

### Examples

```r
GeneList <- c("ALK", "JAK3", "SHC3","TP53","MYC","PARP")
```

```r
## Not run:
cgd <- CGDS("http://www.cbiomedical.org/")
listCase_gbm_tcga_pub <- getCaseLists.CGDS(cgd, "gbm_tcga_pub")[,1]
listGenProf_gbm_tcga_pub <- getGeneticProfiles.CGDS(cgd, "gbm_tcga_pub")[,1]
ProfData_Mut <- grepRef("gbm_tcga_pub_all", listCase_gbm_tcga_pub,
                       "gbm_tcga_pub_mutations", listGenProf_gbm_tcga_pub, GeneList, Mut=1)
```
## End(Not run)
mapLists

Replaces contents of list A with elements of list B

Description

Combines two lists, A and B, such that names(A) are preserved, mapping to the values of B, using names(B) as look up. I.e. replaces values in A with values in B, using names(B) as look up for values in A. Once more? See examples. NB! None-mapped entries are returned as NA, but can be removed using removeNAs.

Usage

mapLists(A, B, removeNAs = TRUE)

Arguments

A              List, elements are coerced to character for mapping to B.
B              List.
removeNAs      Boolean, whether to remove the NAs that occur because an element was not found in B.

Value

List.

Author(s)

Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>

See Also

removeNAs

Examples

A <- list('a1'='alpha', 'a2'='beta', 'a3'=c('gamma','delta'))
B <- list('alpha'='b1', 'gamma'=c('b2', 'b3'), 'delta'='b4')
mapLists(A, B)
metabologram

Circular plot of hierarchital data of genetic profile.

Description

Circular plot of hierarchital data of genetic profile.

Usage

metabologram(treeData, width=600, height=600, main="", showLegend=FALSE, 
legendBreaks=NULL, 
legendColors=NULL, 
fontSize=12, 
legendText="Legend")

Arguments

- **treeData**: A hierarchical tree data as in example
- **width**: 600
- **height**: 600
- **main**: Title
- **showLegend**: FALSE
- **legendBreaks**: NULL
- **legendColors**: NULL
- **fontSize**: 12
- **legendText**: Legend

Value

A circular layout with genetic profile.

See Also

https://github.com/armish/metabologram

Examples

```r
How <- "runManually"
## Not run:
metabologram(treeData = sampleWheelData, width=600, 
height=600, main="title", showLegend = TRUE, fontSize = 10, 
legendBreaks=c("NA","Min","Negative", "0", "Positive", "Max"), 
legendColors=c("black","blue","cyan","white","yellow","red"), 
legendText="Legend")
## End(Not run)
```
**metabologramOutput**  
*Widget output function for use in Shiny*

**Description**  
Widget output function for use in Shiny

**Usage**

metabologramOutput(outputId, width = 600, height = 500)

**Arguments**

- **outputId**: id
- **width**: 600
- **height**: 600

**Value**

A circular plot with genetic profile in Shiny App.

**Examples**

```r
## Not run:
library(bioCancer)
bioCancer::metabologram(treeData = sampleMetabologramData)
## End(Not run)
```

**Mutation_obj**  
*Attribute mutation frequency to nodes*

**Description**  
Attribute mutation frequency to nodes

**Usage**

Mutation_obj(list, FreqMutThreshold, geneListLabel)

**Arguments**

- **list**: A list of data frame with mutation data. Each data frame to study
- **FreqMutThreshold**: threshold Rate of cases (patients) having mutation (0-1).
- **geneListLabel**: file name of geneList examples: "73"
Node_df_FreqIn

Attributes size to Nodes depending on number of interaction

Description

Attributes size to Nodes depending on number of interaction

Usage

Node_df_FreqIn(genelist, freqIn)

Arguments

genelist a vector of genes
freqIn dataframe with Node interaction frequencies

Value

A data frame with nodes size attributes

Examples

Node_df_FreqIn

## Not run:
r_data <- new.env()
GeneList <- whichGeneList("DNA_damage_Response")
node_df <- Node_df_FreqIn(GeneList, r_data$FreqIn)
Node_Diseases_obj  Attributes color and shape to Nodes of Diseases

Description

Attributes color and shape to Nodes of Diseases

Usage

Node_Diseases_obj(genesclassdetails)

Arguments

genesclassdetails

a dataframe from geNetClassifier function

Value

A data frame with nodes Shapes and colors

Examples

GenesClassDetails <- structure(list(Genes = c("FANCF", "MLH1", "MSH2", "ATR", "PARP1", "CHEK2", "RAD51"), ranking = c(1L, 1L, 1L, 2L, 3L, 1L, 2L), class = c("brca_tcga", "gbm_tcga", "lihc_tcga", "lihc_tcga", "lusc_tcga", "lusc_tcga"), postProb = c(1, 0.99, 1, 0.99, 0.99, 1, 0.98), exprsMeanDiff = c(180, 256, -373, -268, -1482, 258, 143), exprsUpDw = c("UP", "UP", "DOWN", "DOWN", "UP", "UP"), .Names = c("Genes", "ranking", "class", "postProb", "exprsMeanDiff", "exprsUpDw"), class = "data.frame", row.names = c(NA,-7L))
Node_Diseases_df <- Node_Diseases_obj(genesclassdetails= GenesClassDetails)

Node_obj_CNA_ProfData  Attribute CNA data to node border

Description

Attribute CNA data to node border

Usage

Node_obj_CNA_ProfData(list)
Node_obj_FreqIn

Arguments

list A list of data frame with CNA data. Each data frame corresponds to a study.

Value

A data frame with node border attributes

Examples

## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
GeneList <- whichGeneList("DNA_damage_Response")
ProfDataCNA <- getProfileData.CGDS(cgds, GeneList, "brca_tcga_pub_gistic","brca_tcga_pub_all")
ListProfDataCNA <- list(ls1=ProfDataCNA, ls2=ProfDataCNA)
nodeObj <- Node_obj_CNA_ProfData(ListProfDataCNA)
## End(Not run)

Node_obj_FreqIn

Attribute interaction frequency to node size

Description

Attribute interaction frequency to node size

Usage

Node_obj_FreqIn(geneList)

Arguments

geneList A list of gene symbol

Value

A data frame with node attributes

Examples

r_data <- new.env()
r_data[["FreqIn"]]<-structure(list(Genes = c("ATM", "ATR", "BRCA1", "BRCA2", "CHEK1", "CHEK2", "FANCF", "MDC1", "RAD51"), FreqSum = c(0.04, 0.05, 0.05, 0.03, 0.05, 0.04, 0.03, 0.03, 0.02)), .Names = c("Genes", "FreqSum"), class = "data.frame", row.names = c(NA, -9L))
## Not run:
GeneList <- whichGeneList("DNA_damage_Response")
nodeObj <- Node_obj_FreqIn(GeneList)
## End(Not run)
**Node_obj_Met_ProfData**  
*Attribute gene Methylation to Nodes*

**Description**

Attribute gene Methylation to Nodes

**Usage**

```r
Node_obj_Met_ProfData(list, type, threshold)
```

**Arguments**

- `list` : a list of data frame with methylation data
- `type` : HM450 or HM27
- `threshold` : the Rate cases (patients) that have a silencing genes by methylation

**Value**

a data frame with node shape attributes

**Examples**

```r
## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
GeneList <- whichGeneList("DNA_damage_Response")
ProfDataMET <- getProfileData(cgds, GeneList, "gbm_tcga_pub_methylation","gbm_tcga_pub_all")
ListProfDataMET <- list(ls1=ProfDataMET, ls2=ProfDataMET)
nodeObj <- Node_obj_Met_ProfData(ListProfDataMET, "HM450",0.1)
## End(Not run)
```

---

**Node_obj_mRNA_Classifier**  
*Attribute genes expression to color nodes*

**Description**

Attribute genes expression to color nodes

**Usage**

```r
Node_obj_mRNA_Classifier(geneList,genesClassDetails)
```
Arguments

geneList A gene list.
genesclassdetails A dataframe with genes classes and genes expression.

Value

A data frame with node color attributes

Examples

r_data <- new.env()
input <- NULL

r_data[["FreqIn"]]<- structure(list(Genes = c("ATM", "ATR", "BRCA1", "BRCA2", "CHEK1", "CHEK2", "FANCF", "MDC1", "RAD51"), FreqSum = c(0.04, 0.05, 0.05, 0.03, 0.05, 0.04, 0.03, 0.03, 0.02)), .Names = c("Genes", "FreqSum"), class = "data.frame", row.names = c(NA, -9L))

GenesClassDetails <- structure(list(Genes = c("FANCF", "MLH1", "MSH2", "ATR", "PARP1", "CHEK2", "RAD51"), ranking = c(1L, 1L, 1L, 2L, 3L, 1L, 2L), class = c("brca_tcga", "gbm_tcga", "lihc_tcga", "lihc_tcga", "lusc_tcga", "lusc_tcga"), postProb = c(1, 0.99, 1, 0.99, 0.99, 1, 0.98), exprsMeanDiff = c(180, 256, -373, -268, -1482, 258, 143), exprsUpDw = c("UP", "UP", "DOWN", "DOWN", "UP", "UP")), .Names = c("Genes", "ranking", "class", "postProb", "exprsMeanDiff", "exprsUpDw"), class = "data.frame", row.names = c(NA,-7L))

## Not run:
GeneList <- whichGeneList("DNA_damage_Response")
nodeObj <- Node_obj_mRNA_Classifier(GeneList, GenesClassDetails)

## End(Not run)

---

pickGO

Cleans up result from org.Xx.egGO and returns specific GO identifiers

Description

Cleans up result from org.Xx.egGO and returns GO identifier for either biological process (BP), cellular component (CC), or molecular function (MF). Can be used on list of GOs from translate, or a single list of GOs from an annotation package. May reduce list, if the (sub)list does not contain the chosen class!

Usage

pickGO(l, evidence = NA, category = NA)
pickGO

Arguments

- **1** Character vector, or list of GO identifiers.
- **evidence** Character vector, filters on which kind of evidence to return; for a larger list see `getEvidenceCodes`. ▲ Evidence codes may be: c('IMP', 'IGI', 'IPI', 'ISS', 'IDA', 'IEP', 'IEA', 'TAS', 'NAS', 'ND', 'IC'). ▲ Leave as NA to ignore filtering on this part.
- **category** Character vector, filters on which ontology to return: biological process (BP), cellular component (CC), or molecular function (MF). ▲ Leave as NA to ignore filtering on this part.

Value

List with only the picked elements.

Author(s)

Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>

See Also

`pickRefSeq`, `getEvidenceCodes`, `translate`

Examples

```r
library(org.Bt.eg.db)
genes <- c(280705, 280706, 100327208)
translate(genes, org.Bt.egSYMBOL)

symbols <- c("SERPINA1","KERA","CD5")
refseq <- translate(symbols, from=org.Bt.egSYMBOL2EG, to=org.Bt.egREFSEQ)
# Pick the proteins:
pickRefSeq(refseq, priorities=c('NP','XP'), reduce='all')

# If you wanted do some further mapping on the result from
# translate, simply use lapply.

library(GO.db)
GO <- translate(genes, org.Bt.egGO)
# Get all biological processes:
## Not run:
pickGO(GO, category='BP')
  # $ 280705
  # [1] "GO:0006826" "GO:0006879"
  # $ 280706
  # [1] "GO:0006590" "GO:0007165" "GO:0042446"

# Get all ontologies with experimental evidence:
pickGO(GO, evidence=c('IMP', 'IGI', 'IPI', 'ISS', 'IDA', 'IEP', 'IEA'))
  # $ 280705
  # [1] "GO:0006826" "GO:0006879" "GO:0005615" "GO:0008199"
  # $ 280706
  # [1] "GO:0006590" "GO:0007165" "GO:0042446" "GO:0005615" "GO:0005179" "GO:0042393"
```
pickRefSeq

## End(Not run)

---

**pickRefSeq**

*Picks a prioritised RefSeq identifier from a list of identifiers*

### Description

When translating to RefSeq, typically multiple identifiers are returned, referring to different types of products, such as genomic molecule, mature mRNA or the protein, and they can be predicted, properties that can be read from the prefix ([https://www.ncbi.nlm.nih.gov/refseq/](https://www.ncbi.nlm.nih.gov/refseq/)). E.g. "XM_" is predicted mRNA and "NP_" is a protein. Run `?org.Bt.egREFSEQ`.

### Usage

```r
pickRefSeq(
  l,
  priorities = c("NP", "XP", "NM", "XM"),
  reduce = c("all", "first", "last")
)
```

### Arguments

- **l**: Vector or list of RefSeqs accessions to pick from. If list given, applies the prioritization to each element in the list.
- **priorities**: Character vector of prioritised prefixes to pick by. E.g. `c("NP","NM")` returns RefSeqs starting 'NP', and if none found, those starting 'NM'. If no RefSeqs are found according to the priorities, Null is returned, unless the last element in priorities is '*'. Uses `grepl`, so see these for pattern matching. Default: `c("NP","XP","NM","XM")`
- **reduce**: Reducing method, either return all annotations (one-to-many relation) or the first or last found annotation. The reducing step is applied after translating to the goal: all: returns all annotations first or last: choose first or last of arbitrarily ordered list.

### Value

If vector given, returns vector. If list given, returns list without element where nothing could be picked.

### Author(s)

Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>
Examples

```r
library(org.Bt.eg.db)
symbols <- c("SERPINA1", "KERA", "CDS")
refseq <- translate(symbols, from=org.Bt.egSYMBOL2EG, to=org.Bt.egREFSEQ)
mRNA <- pickRefSeq(refseq, priorities=c("NM","XM"))
proteins <- pickRefSeq(refseq, priorities=c("NP","XP"))
```

Description

These methods should not be invoked by the user.

Usage

```r
## S3 method for class 'CGDS'
processURL(x, url, force.comment.char.blank = FALSE, ...)
```

Arguments

- `x`: A connection object
- `url`: URL
- `force.comment.char.blank`: a boolean param to force comment
- `...`: not used

removeNAs

Removes entries equal NA from list or vector

Description

Removes entries equal NA, but not mixed entries containing, amongst others, NA. Good for use after `mapLists` that might return entries equal NA.

Usage

```r
removeNAs(l)
```

Arguments

- `l`: Vector or list.

Author(s)

Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>
**Examples**

```r
removeNAs(list('a'=NA, 'b'=c(NA, 'B'), 'c'='C'))
```

**Description**

Widget render function for use in Shiny

**Usage**

```r
renderCoffeewheel(expr, env = parent.frame(), quoted = FALSE)
```

**Arguments**

- `expr`: expression
- `env`: environment
- `quoted`: logical

**Value**


**Examples**

```r
How <- "runManually"
## Not run:
coffeewheel(treeData = sampleWheelData)
## End(Not run)
```

**Description**

Widget render function for use in Shiny

**Usage**

```r
renderMetabologram(expr, env = parent.frame(), quoted = FALSE)
```
Arguments

expr  expression
env   parent.frame()
quoted  FALSE

Value

A circular plot with genetic profile in Shiny App.

Examples

## Not run:
library(bioCancer)
bioCancer::metabologram(treeData = sampleMetabologramData)

## End(Not run)

---

reStrColorGene  
Restructure the list of color attributed to the genes in every dimension for every studies

Description

Restructure the list of color attributed to the genes in every dimension for every studies

Usage

reStrColorGene(df)

Arguments

df  data frame with colors attributed to the genes

Value

Hierarchical color attribute: gene > color

Examples

## Not run:
cgds <- CGDS("http://www.cbiportal.org/")
geneList <- whichGeneList("73")
ProfData <- getProfileData.CGDS(cgds,
   geneList, "gbm_tcgag_pub_mrna", "gbm_tcgag_pub_all")
rownames(ProfData) <- NULL
ls <- reStrColorGene(ProfData)

## End(Not run)
reStrDimension | Restructure the list of color attributed to the genes in every study for every dimensions

**Description**
Restructure the list of color attributed to the genes in every study for every dimensions

**Usage**
reStrDimension(LIST)

**Arguments**
LIST list of hierarchical dimensions

**Value**
Hierarchical structure of: Study > dimensions > gene > color

**Examples**
```r
## Not run:
cgdS <- CGDS("http://www.cbiportal.org/")
genList <- whichGeneList("73")
ProfData <- getProfileData.CGDS(cgdS,
genList, "gbm_tcgA_pub_mrna", "gbm_tcgA_pub_all")
rownames(ProfData) <- NULL
TREE <- reStrDimension(list(
  list1=list(df1=ProfData,df2=ProfData),
  list2=list(df3=ProfData,df4=ProfData)))

## End(Not run)
```

---

reStrDisease | Restructure the list of color attributed to the genes in every disease

**Description**
Restructure the list of color attributed to the genes in every disease

**Usage**
reStrDisease(List)

**Arguments**
List of data frame with color attributes
Value

Hierarchy of dimensions in the same study: dimensions > gene > color

Examples

```r
## Not run:
library(CGDS)
library(ggplot2)
library(reshape2)
library(dplyr)
library(tidyverse)

# Create a data frame with expression values
expression_data <- data.frame(
  gene = c("GeneA", "GeneB", "GeneC"),
  condition = c("Condition1", "Condition2", "Condition3"),
  expression = c(100, 200, 300),
  name = c("GeneA", "GeneB", "GeneC")
)

# Create a data frame with clinical data
clinical_data <- data.frame(
  patient = c("Patient1", "Patient2", "Patient3"),
  diagnosis = c("Diagnosis1", "Diagnosis2", "Diagnosis3"),
  stage = c("Stage1", "Stage2", "Stage3"),
  name = c("Patient1", "Patient2", "Patient3")
)

# Merge expression and clinical data
merged_data <- merge(expression_data, clinical_data, by="name")
```

Description

Return message when the filter formula is not correct (mRNA > 500)

Usage

```r
returnTextAreaInput(inputId, label= NULL, rows = 2, placeholder = NULL, resize= "vertical", value = "")
```

Arguments

- **inputId**: The ID of the object
- **label**: Text describes the box area
- **rows**: Number of rows
- **placeholder**: Error message if needed
- **resize**: orientation of text
- **value**: default text in the area box

Value

text message
Examples

```r
ShinyApp <- 1
## Not run:
returnTextAreaInput(inputId = "data-filter",
label = "Error message",
rows = 2,
placeholder = "Provide a filter (e.g., Genes == 'ATM') and press return",
resize = "vertical",
value="")
## End(Not run)
```

Description

Sets verbose logging level for CGDS function calls.

Usage

```r
## S3 method for class 'CGDS'
setVerbose(x, verbose, ...)
```

Arguments

- `x` A connection object
- `verbose` Activate verbose logging (boolean)
- `...` not used

Examples

```r
# Create CGDS object
mycgds <- CGDS("http://www.cbioportal.org/")
# Activate verbose logging
setVerbose.CGDS(mycgds, TRUE)
```
### switchButton

A function to change the Original checkbox of shiny into a nice true/false or on/off switch button No javascript involved. Only CSS code.

#### Description

To be used with CSS script 'button.css' stored in a `www` folder in your Shiny app folder

#### Usage

```r
switchButton(inputId, label = NULL, value = FALSE, col = "GB", type = "TF")
```
Arguments

- **inputId**: The input slot that will be used to access the value.
- **label**: Display label for the control, or NULL for no label.
- **value**: Initial value (TRUE or FALSE).
- **col**: Color set of the switch button. Choose between "GB" (Grey-Blue) and "RG" (Red-Green).
- **type**: Text type of the button. Choose between "TF" (TRUE - FALSE), "OO" (ON - OFF) or leave empty for no text.

__Description__

S3 method to test cBioPortal connection

__Usage__

```r
## S3 method for class 'CGDS'
test(x, ...)
```

__Arguments__

- **x**: connection object
- **...**: not used

__translate__

Translate between different identifiers

__Description__

Function for translating from one annotation to another, eg. from RefSeq to Ensemble. This function takes a vector of annotation values and translates first to the primary annotation in the Biocore Data Team package (ie. entrez gene identifier for org.Bt.eg.db) and then to the desired product, while removing non-translated annotations and optionally reducing the result so there is only a one-to-one relation.
translate

translate(
  values,
  from,
  to = NULL,
  reduce = c("all", "first", "last"),
  return.list = TRUE,
  remove.missing = TRUE,
  simplify = FALSE,
  ...
)

Arguments

values Vector of annotations that needs translation. Coerced to character vector.
from Type of annotation values are given in. NB! take care in the orientation of the package, ie. if you have RefSeq annotations, use org.Bt.egREFSEQ2EG or (in some cases) revmap(org.Bt.egREFSEQ).
to Desired goal, eg. org.Bt.egENSEMBLPROT. If NULL (default), goal if the packages primary annotation (eg. entrez gene for org.Bt.eg.db). Throws a warning if the organisms in from and to are not the same.
reduce Reducing method, either return all annotations (one-to-many relation) or the first or last found annotation. The reducing step is applied after translating to the goal: all: returns all annotations first or last: choose first or last of arbitrarily ordered list.
return.list Logical, when TRUE, returns the translation as a list where names
remove.missing Logical, whether to remove non-translated values, defaults TRUE.
simplify Logical, unlists the result. Defaults to FALSE. Usefull when using translate in a lapply or sapply.
... Additional arguments sent to pickGO if from returns GO set.

Details

If you want to do some further mapping on the result, you will have to use either unlist og lapply, where the first returns all the end-products of the first mapping, returning a new list, and the latter produces a list-within-list.

If from returns GO identifiers (e.g. from = org.Bt.egGO), then the returned resultset is more complex and consists of several layers of lists instead of the usual list of character vectors. If to has also been specified, the GO IDs must be extracted (internally) and you have the option of filtering for evidence and category at this point. See pickGO.

Value

List; names of elements are values and the elements are the translated elements, or NULL if not translatable with remove.missing = TRUE.
UnifyRowNames

Note

Requires user to deliver the annotation packages such as org.Bt.egREFSEQ.

Author(s)

Stefan McKinnon Edwards <stefan.hoj-edwards@agrsci.dk>

See Also

pickRefSeq, pickGO

Examples

```r
library(org.Bt.eg.db)
genes <- c(280705, 280706, 100327208)
translate(genes, org.Bt.egSYMBOL)

symbols <- c("SERPINA1", "KERA", "CDS")
refseq <- translate(symbols, from=org.Bt.egSYMBOL2EG, to=org.Bt.egREFSEQ)

# Pick the proteins:
pickRefSeq(refseq, priorities=c('NP', 'XP'), reduce='all')

# If you wanted do do some further mapping on the result from
# translate, simply use lapply.

library(GO.db)
GO <- translate(genes, org.Bt.egGO)
```

UnifyRowNames

Unify row names in data frame with the same order of gene list.

Description

Unify row names in data frame with the same order of gene list.

Usage

`UnifyRowNames(x, geneList)`

Arguments

- `x` data frame with gene symbol in the row name
- `geneList` a gene list

Value

a data frame having the gene in row name ordered as in gene list.
Examples

```r
## Not run:
cgds <- CGDS("http://www.cbioportal.org/")
geneList <- whichGeneList("73")
ProfData <- getProfileData.CGDS(cgds,
genelist, "gbm_tcga_pub_mrna", "gbm_tcga_pub_all")
rownames(ProfData) <- NULL
geneListOrder <- UnifyRowNames(list(
  list1=list(df1=ProfData, df2=ProfData),
  list2=list(df3=ProfData, df4=ProfData),
genelist))

## End(Not run)
```

---

user_CNA  | Example of Copy Number Alteration (CNA) dataset

Description

Example of Copy Number Alteration (CNA) dataset

Usage

```r
user_CNA
```

Format

An object of class `data.frame` with 579 rows and 13 columns.

Author(s)

Karim Mezhoud <kmezhoud@gmail.com>

---

user_MetHM27  | Example of Methylation HM27 dataset

Description

Example of Methylation HM27 dataset

Usage

```r
user_MetHM27
```

Format

An object of class `data.frame` with 600 rows and 13 columns.
**user_MetHM450**

**Description**
Example of Methylation HM450 dataset

**Usage**
user_MetHM450

**Format**
An object of class `data.frame` with 10 rows and 13 columns.

**Author(s)**
Karim Mezhoud <kmezhood@gmail.com>

**user_mRNA**

**Description**
Example of mRNA expression dataset

**Usage**
user_mRNA

**Format**
An object of class `data.frame` with 307 rows and 13 columns.

**Author(s)**
Karim Mezhoud <kmezhood@gmail.com>
**user_Mut**

*Example of Mutation dataset*

**Description**

Example of Mutation dataset

**Usage**

`user_Mut`

**Format**

An object of class `data.frame` with 37 rows and 23 columns.

**Author(s)**

Karim Mezhoud &lt;kmezhoud@gmail.com&gt;

---

**whichGeneList**

*Verify which gene list is selected*

**Description**

Verify which gene list is selected

**Usage**

`whichGeneList(geneListLabel)`

**Arguments**

- `geneListLabel` The label of GeneList. There are three cases: "Genes" user gene list, "Reactome_GeneList" GeneList plus genes from reactomeFI "file name" from Examples

**Value**

Gene List label

**Examples**

```r
How &lt;- "runManually"
## Not run:
whichGeneList("102")
## End(Not run)
```
**Description**

Capture html output widget as .png in R

**Usage**

```r
widgetThumbnail(p, thumbName, width = 1024, height = 1024)
```

**Arguments**

- `p` is the html widget
- `thumbName` is the name of the new png file
- `width` 1024
- `height` 1024

**Value**

3 files .html, .js and .png

**Examples**

```r
How <- "runManually"
## Not run:
# Load package
library(networkD3)
library(htmlwidgets)
# Create fake data
networkData <- data.frame(src, target)
# Plot
plot = simpleNetwork(networkData)
# Save html as png
widgetThumbnail(p = plot, thumbName = "plot", width = 1024, height = 1024)
## End(Not run)
```
Index

* datasets
  - epiGenomics, 14
  - user_CNA, 48
  - user_MethM27, 48
  - user_MethM450, 49
  - user_mRNA, 49
  - user_Mut, 50

* package
  - AnnotationFuncs-package, 3
  - .dbEscapeString, 4
  - .getTableName, 5, 25
  - .pickRef, 5
  - AnnotationFuncs
    - (AnnotationFuncs-package, 3
    - AnnotationFuncs-package, 3
  - attriColorGene, 6
  - attriColorValue, 7
  - attriColorVector, 8
  - attriShape2Gene, 8
  - attriShape2Node, 9
  - bioCancer, 10
  - CGDS, 10
  - checkDimensions, 11
  - coffeewheel, 11
  - coffeewheelOutput, 12
  - displayTable, 13
  - Edges_Diseases_obj, 13
  - epiGenomics, 14
  - findPhantom, 15
  - getGenesClassification, 18
  - getGeneticProfiles.CGDS, 19
  - getList_Cases, 21
  - getList_GenProfs, 21
  - getListProfData, 20
  - getMegaProfData, 22
  - getMutationData.CGDS, 23
  - getOrthologs, 3, 4, 24
  - getProfileData.CGDS, 25
  - getSequenced_SampleSize, 26
  - grepRef, 27
  - mapLists, 25, 28, 38
  - metabologram, 29
  - metabologramOutput, 30
  - Mutation_obj, 30
  - Node_df_FreqIn, 31
  - Node_Diseases_obj, 32
  - Node_obj_CNA_ProfData, 32
  - Node_obj_FreqIn, 33
  - Node_obj_Met_ProfData, 34
  - Node_obj_mRNA_Classifier, 34
  - pickGO, 17, 35, 46, 47
  - pickRefSeq, 5, 6, 36, 37, 47
  - processURL.CGDS, 38
  - removeNAs, 28, 38
  - renderCoffeewheel, 39
  - renderMetabologram, 39
  - reStrColorGene, 40
  - reStrDimension, 41
  - reStrDisease, 41
  - returnTextAreaInput, 42
  - setVerbose.CGDS, 43
  - Studies_obj, 44
  - switchButton, 44
  - test.CGDS, 45
translate, 3, 4, 24, 25, 35, 36, 45

UnifyRowNames, 47
user_CNA, 48
user_MethM27, 48
user_MethM450, 49
user_mRNA, 49
user_Mut, 50

whichGeneList, 50
widgetThumbnail, 51